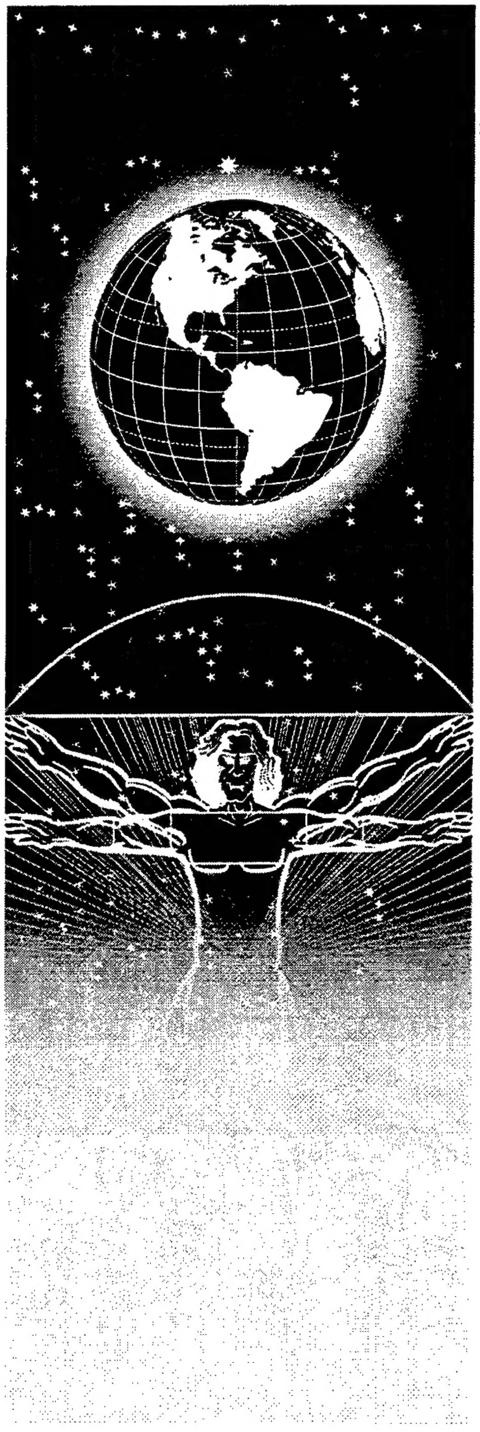
AL/OE-TR-1996-0102



UNITED STATES AIR FORCE
ARMSTRONG LABORATORY

A Comparison Of Various Probit Methods For Analyzing Yes/No Data On A Log Scale

Clarence P. Cain
Gary D. Noojin

TASC
4241 Woodcock Drive
Suite B100
San Antonio, TX 78228

Lonnie Manning, Captain, USAF

December 1996

Occupational and Environmental
Health Directorate
Optical Radiation Division
8111 18th Street
Brooks Air Force Base, TX 78235-5215

[illegible]

ROBERT M. CARTLEDGE, Lt Col, USAF, BSC
Chief, Optical Radiation Division

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE December 1996	3. REPORT TYPE AND DATES COVERED Interim - Sep 94 - Sep 95

4. TITLE AND SUBTITLE A Comparison of Various Probit Methods for Analyzing Yes/No Data on a Log Scale	5. FUNDING NUMBERS C - F33615-92-C-0017 PE - 62202F PR - 2312 TA - A1 WU - 01
6. AUTHOR(S) Clarence P. Cain, Gary D. Noojin, and Lonnie Manning	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) TASC 4241 Woodcock Drive Suite B100 San Antonio, TX 78228	8. PERFORMING ORGANIZATION
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Armstrong Laboratory Occupational and Environmental Health Directorate Optical Radiation Division 8111 18th Street Brooks AFB, TX 78235-5215	10. SPONSORING/MONITORING AL/OE-TR-1996-0102
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A study of the various probit methods to analyze biological data was undertaken to understand the various methods and to determine the requirements for the input data as to distribution, number, and tightness of data for the desired results. Calculations were run for many data sets, and results were compared. Graphical analysis and the Kärber method were used along with the SAS Probit Procedure and EZ-Probit. All four methods provided very close agreement on most data sets, and the EZ-Probit program provided almost identical information to the SAS Probit Procedure. Real biological data sets were used for comparison purposes, and three other data sets were made up to simulate real data (with variations in the number and distribution of data points). Fiducial limits at the 95% confidence level were also calculated and compared. For those data sets which had no fiducial limits with SAS Probit at the 95% confidence interval, 85% confidence levels were calculated with the EZ-Probit method because it is not possible to adjust the confidence levels with SAS Probit. Different data sets were run in an attempt to minimize the amount of additional data points needed for an existing data set to tighten the fiducial limits and to show the correlation with a chi-square distribution. Finally, it will be shown that the probability distributions from 0.01 to 0.99 are identical out to four decimal places for the SAS Probit and EZ-Probit and that there are only minor differences in the fiducial limits between the two methods for some data sets. Appendix A contains the program required to run the SAS Probit on a PC computer. The EZ-Probit method is included in Appendix B.

14. SUBJECT TERMS Biological data; Calculations; Fiducial limits; Probit			15. NUMBER OF PAGES 46 16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev 2-89) Prescribed by ANSI Std Z-39-18
298-102 COMPUTER GENERATED

TABLE OF CONTENTS

LIST OF TABLES

INTRODUCTION

Probit analysis was developed to analyze discrete or dichotomous data obtained by many research endeavors including natural or threshold response rate for biological systems. We have been using this method as a statistical tool to determine the probability of dose-response curves for minimum visible lesions (MVLs) produced within the eye for single laser pulses. In most cases, the dose or laser pulse energy values required to produce a visible lesion within the eye are reported as the ED_{50} dose or that dose which has a probability of 50% of creating a visible lesion. However, the complete probability curve is calculated during the iterative process, and the printout generally gives points between 1% and 99%.

The beginning of probit analysis dates back over 60 years when the Kärber[1] method was first published to estimate the mean and standard deviation of the distribution of data where the distribution was unknown. The Kärber method can still be used as a first approximation for the nonparametric estimates of the mean and standard deviation for the exact probit solution which involves a series of successive approximations. The exact probit method of solution has replaced most other methods because it conveys the maximum amount of information from the data. In addition, the exact method provides the associated confidence intervals, which allows inferences to be made about the quality of the experiment and tightness of the data so that obvious deficiencies may be corrected.

Graphical methods[2,3] for the probit analysis have been around for over 50 years and were used quite extensively because of their simplicity and ease of use. The data is plotted using this method, a straight line is drawn through the data points, and the ED_{50} level is read from the graph. The slope of the straight line can also be determined from the graph. Later, a simplified graphical method was developed which included graph paper especially designed to permit rapid analysis. However, all of these methods were developed before the modern day computer and therefore cannot compete with the speed or accuracy of computer-derived exact solutions of the probit iterative process. The bootstrap method[4] was not considered in this study because we were interested in only two-level trials (i.e., yes/no data), and this type of data was not compared in the study by Foster and Bischof.[4]

METHODS

Most of the original theoretical development of the probit methods can be attributed to Finney[3], who first published a book on probit analyses in 1947. Since then, there have been a number of revisions, editions, and reprinting of his original book, and in one 10-year period between 1978 - 1988, there were over 2300 citations[4] of his books in the literature. Thus, his procedure is the most widely used to analyze yes/no data and other discrete event data. Finney developed the mathematical models to utilize probit analyses, but it still takes a computer program to process the data. Many programs for many different computers have been written to perform the probit procedures which were developed by Finney, and most will calculate the probability curve equally well. However, to standardize the reporting of data in the literature, the SAS Institute, Inc.[5], Cary, NC, developed computer programs for many different computer systems and operating systems which calculate maximum-likelihood estimates of regression parameters and natural (threshold) response rate for the discrete event data. Thus, these programs calculate the exact same results for all systems. One of the authors of this report, Captain Lonnie Manning, has written a program in C++ that operates on a PC computer to utilize the methods developed by Finney (see Appendix B). This program produces all of the same output parameters as SAS Probit. The SAS Institute has named its process The Probit Procedure, and we call ours EZ-probit.

There were several reasons for writing the EZ-probit program. In the literature, we find many references to Finney and the method he employs for the Probit calculations. However, SAS's Probit and Finney's Probit are different. Although Finney does briefly mention the method employed by SAS (SAS's Probit is actually a Normit since the Probit is not centered about Y-5). The EZ-probit program incorporates Finney's expressions exactly. Comparison of the SAS output and that from Finney's Probit shows they differ for a small number of data points with agreement increasing as the number of data points increase. This is most noticeable in the fiducial limit calculations. In most instances, the difference is inconsequential compared to the experimental errors inherent in our type of experiments(i.e.+/- 7% for energy).

The Probit Procedure computes maximum-likelihood estimates of the slope and intercept of the probit equation using a modified Newton-Raphson algorithm. The data set used by SAS Probit must include either a response variable giving the level of response for each observation or a pair of variables giving the number of subjects tested and the number of subjects responding for each dose of the independent variable values. Two goodness-of-fit chi-square values are computed if requested. Inverse confidence limits for one of the independent variables can be requested, and the confidence limits are computed using a critical value of 1.96, which corresponds to an approximate 95% confidence interval. If the Pearson goodness-of-fit chi-square test is requested and the p-value for the test is too small, variances and covariances are adjusted by a heterogeneity factor, and a critical value from the t-distribution is used to compute the fiducial limits (FLs). The p-value used for the chi-square test can be set to different levels with a default p-value of 0.10. Also calculated and outputted is the slope of the probit line between the ED_{84} and the ED_{50} values.

The EZ-probit program is faster and much more user-friendly than the existing SAS program. All relevant parameters can be set by the click of a mouse. To use SAS's Probit, the whole SAS program must be loaded. EZ-probit is a Windows application written in C++ (Borland 4.0 Compiler) and comes in both a 16 bit or 32 bit application. In addition, EZ-probit allows flexibility in setting the confidence to something other than 95% which is currently very difficult to accomplish in SAS.

A good treatise on the graphical methods was developed by Frisch.[6] Step-by-step procedures are presented in that report, together with comparisons between graphical, Kärber, and exact probit solution. In this paper, we present his results and compare them to results run on SAS Probit and EZ-Probit, along with results from the MVL and other data sets created to influence their outputs.

DATA SETS

The first data set presented comes from Frisch[6], which is for the 70-millisecond (ms) pulse duration and 170 exposures. In his report, Frisch thoroughly analyzes the three methods and clearly explains each step in each method. Real data are used in each case. His report should be consulted for learning each procedure and what auxiliary data is necessary. Table 1 lists the results from Frisch's calculations for the graphical method, Kärber approximation, his exact probit solution, and the results from SAS Probit and our EZ-Probit (both of which calculate the exact probit solution using an iterative process).

Table 1. Comparison of Four Methods for Data from Frisch.[6]

Parameter	Graphical	Kärber	Exact Probit	SAS Probit	EZ-Probit
ED_{50}	12.5	13.15	12.5	12.4	12.4
Upper FL	14.4	14.55	14.0	13.8	13.8
Lower FL	11.2	11.65	11.0	10.8	10.8
Slope of Probit	5.9	----	5.68	5.67	5.67
Ratio(ED_{84} / ED_{50})	1.48	----	1.50	1.50	1.50

Analysis of Table 1

The listed graphical, Kärber, and exact probit parameter values above were calculations by Frisch, and the SAS Probit was run on the PC version of The Probit Procedure. The exact probit used by Frisch should be equivalent to the SAS probit since they both use the iterative process to achieve a final solution. The differences between the three methods (graphical, Kärber, and exact probit) are minimal, and it is obvious that the three methods give comparable results. SAS Probit and the EZ-Probit also print out the goodness-of-fit test, and for the Pearson chi-square, the value was 2.23 and the Prob>Chi-Sq was 0.97. SAS Probit and EZ-Probit gave identical results out to four decimal places. Therefore, this data set did have a good distribution. However, this is only one data set, and the results of the calculations depend on the data set. One difference between Frisch and the SAS Probit is the way slope is defined and printed out by SAS Probit. Frisch defines slope of the probit as the ratio of ED_{84} / ED_{50}, while SAS Probit defines it as the slope of the straight line of best fit to the data. The two are inversely related, and the slope of the probit may be obtained from the ED_{84} / ED_{50} ratio simply by taking the reciprocal of the logarithm_{10} of the ratio (ED_{84} / ED_{50}). About 2.5 is the crossover point of these two numbers, and the theoretical minimum for the ratio is "1" (i.e., the slope of the probit would be infinite in this case). However, for made-up data, the smallest ratio is 1.02, while the slope is calculated to be 90 (see Table 9).

In order to make a valid comparison, real data sets as well as created data sets must be analyzed using all of the different methods, and their outputs must be compared. Other data sets compared were the MVL data for 90- and 600-femtosecond (fs) pulsewidths. These data sets consisted of 122 exposures at 90 fs and 121 exposures at 600 fs, both at 580-nanometer (nm) wavelength for the rhesus monkey eye. A graphical analysis was performed on each set and compared with SAS Probit results. The results for these two sets are listed in Tables 2 and 3.

Table 2. MVL Data at 90 Femtoseconds.

Parameter	Graphical	Kärber	SAS Probit	EZ-Probit
ED_{50}	0.46	0.60	0.43	0.43
Upper FL	0.77	0.71	0.61	0.61
Lower FL	0.25	0.49	0.27	0.27
Slope of Probit	1.04	----	1.61	1.61
Ratio(ED_{84} / ED_{50})	9.9	----	4.2	4.2

Analysis of Table 2

Table 2 again shows the comparison between graphical, Kärber, and exact probit as calculated by SAS Probit and EZ-Probit. This data set was selected because of its spread in data and its low slope calculated by SAS. This data set illustrates the necessity of the iterative process involving a series of successive approximations to obtain the final solution. The Kärber method gives only a first approximation, while the graphical method gives only an eyeball approximation with its linear regression. SAS Probit calculated a higher slope value than the graphical, 1.61 versus 1.04, respectively. The goodness-of-fit test gave a value of 55 and a Prob>Chi-Sq of 0.92. If one assumes that a good and reasonable data set should have at least a 0.90 for a Prob, then this data set does fit the requirement.

The data set shown in Table 3 was selected because of its high value of slope of the probit line and tightness of data points. The goodness-of-fit test gave a value of 30 with a Prob>Chi-Sq of 1.0000. In this case, the three methods gave very close results which is a good indicator of the power of the interactive process of SAS as compared to the data in Table 2. The other two methods give good agreement in their approximation because of the data set. Not only are the ED_{50}s close between the three methods, but the FLs are also very close to each other. Thus, any of these methods would suffice if all data sets were as regular as this set.

Table 3. MVL Data for 600 Femtoseconds.

Parameter	Graphical	Kärber	SAS Probit	EZ-Probit
ED_{50}	0.24	0.28	0.26	0.26
Upper FL	0.32	0.32	0.31	0.31
Lower FL	0.18	0.24	0.21	0.21
Slope of Probit	3.31	----	4.79	4.79
Ratio(ED_{84} / ED_{50})	2.40	----	1.58	1.58

Analysis of Table 3

Some questions have arisen concerning whether the distribution of data in the set is normal or unknown. Since the Kärber method reportedly does not depend on a known distribution, we attempted to show the differences between the methods by using several known distributions of data. Thus, chi-squared, linear, step, and random distributions were used in the following data sets.

The data presented in Tables 4 and 5 were obtained from the equation $Y = 5.6 + 2x$, which plots as a straight line on linear graph paper. The dosage or trial level point distribution was obtained from a table of chi-square values, and there is no spread in the data. This normal chi-square distribution of 90 doses or levels with linear increasing yes responses is presented in Table 4, and the same distribution with varying number of data points is presented in Table 5.

Table 4. Comparison of Three Methods for an Exact Straight Line Distribution. (Ninety doses with chi-square distribution of levels.)

Parameter	Graphical	Kärber	SAS Probit	EZ-Probit
ED_{50}	0.50	0.50	0.50	0.50
Upper FL	0.70	0.72	0.72	0.72
Lower FL	0.36	0.40	0.35	0.35
Slope of Probit	2.0	----	2.0	2.0
Ratio(ED_{84} / ED_{50})	3.2	----	3.2	3.2

Analysis of Table 4

This data set was chosen to see how well SAS Probit performed as compared to the graphical procedure since this is a perfect data set with chi-square distribution and no spread. In this case, the graphical procedure should give an exact solution and the Kärber method could also be compared. The goodness-of-fit test gave a value of 0.0021 with a Prob of 1.0000. As can been seen from the results in the table, there is very good agreement between the four methods. Thus, for this type of data all three methods perform equally well.

Table 5. Same Distribution as Table 4 for Different Number of Data Points. (All calculations with SAS Probit. EZ-Probit values shown in ( ).)

Parameter (# pts)	180	90	45	27	18	14
ED_{50}	0.5	0.5	0.49	0.53	0.50	0.50
Upper FL	0.64	0.72	0.74	1.14	1.45 (1.32)	9.54 (1.67)
Lower FL	0.39	0.35	0.32	0.27	0.17 (.19)	0.02 (.15)
Slope of Probit	2.0	2.0	2.8	2.4	2.8	5.3
Ratio(ED_{84} / ED_{50})	3.3	3.3	2.3	2.9	2.4	1.44
ED_{99}	7.31	7.31	3.35	4.92	3.43	1.38
ED_{01}	0.03	0.03	0.07	0.06	0.07	0.18
G-o-F Value	0.0042	0.0021	1.25	3.39	2.67	1.6
Prob>Chi-Sq	1.00	1.00	0.99	0.85	0.91	0.98

Analysis of Table 5

The data sets presented in Table 5 were all run with SAS Probit and EZ-Probit to show the effects of sample size on the various calculated parameters. EZ-Probit gave identical results to SAS Probit with the exceptions of the two values for the fiducial limits shown in parentheses below each value. Since the EZ-probit program incorporates Finney's expressions exactly, there is slight differences between the two methods for calculating the fiducial limits as shown by these small differences. In most instances however, the differences are inconsequential compared to the experimental errors inherent in our type of experiments(i.e.+/- 7% for energy). The variations in the ED_{50} values were due to the inability to maintain the exact percentages of yes responses in each level because of so few points at each level. With the fewer points (below 45), the percentages at each level were no longer constant and there was spread in the data set so that not all points fell on the straight line. From this table, it can be seen that the midpoint, ED_{50}, remains fairly constant while the FLs spread apart as the number of points decreases. In fact, below 14 data points, neither program could obtain FLs with a

95% confidence level. However, it is possible to run EZ-Probit at any confidence level specified. It is clear that the slope of the probit changed because of the fewer number of "1"s below 0.5 and the ones used were closer to the ED_{50} point (i.e., smaller spread in the data). The span between the ED_{01} and ED_{99} points decreased as the number of points decreased because of the smaller spread in data. From these data sets, it appears that "reasonable" FLs are not obtained until the data set is larger than 27. Herein we define reasonable to be when the upper FL is no larger than 50% of ED_{50}, and the lower FL is no smaller than 50% of ED_{50}.

The next data set (Table 6) to be compared by the three methods was a set of ten dosage levels, equally spaced, with a linear increase from 0 to 100% yes responses for the levels. The levels increased from 0.1 to 1.0 in 0.1 increments, and the response increased proportionally.

Table 6. Comparison for Equal Doses, Equally Spaced and Linear Response Levels by Four Probit Methods for 95 Trials. SAS Probit and EZ-Probit equal to four decimals.

Parameter	Graphical	Kärber	SAS Probit	EZ-Probit
ED_{50}	0.46	0.50	0.50	0.50
Upper FL	0.63	0.56	0.60	0.60
Lower FL	0.40	0.44	0.41	0.41
Slope of Probit	4.0	----	4.0	4.0
Ratio(ED_{84} / ED_{50})	1.81	----	1.91	1.91
ED_{99}			1.91	1.91
ED_{01}			0.13	0.13
G-o-F Value			2.13	2.13
Prob>Chi-Sq			0.98	0.98

Analysis of Table 6

By comparing these four results, all methods work equally well on this data set. However, a straight line was drawn through a severely curved set of data points, and it was not obvious by looking at the graph that it would give correct results. In this case, eyeballing gave results as good as any of the other three methods. Thus, equally spaced dosages or trial levels appear to be as good as or better than chi-square distribution of levels. The goodness-of-fit test gave a value of 2.13, and the Prob was 0.98 for both methods run on the computer.

The next set of calculations was performed on the same distribution but with a decreasing number of samples. The same percentages of yes responses were maintained to their limit. These calculations were all performed with SAS Probit and EZ-Probit (Table 7), and the sample size was reduced from 95 data points to 12. EZ-Probit values, if different from SAS Probit values, are shown in parentheses. Note that where SAS Probit was not able to calculate the FLs at the 95% confidence level for the 12 data points, EZ-Probit was able to calculate the limits at the 85% confidence level.

Table 7. A Comparison of SAS Probit Calculations for Data Sets with Decreasing Number of Data Points. (Same data as in Table 6.)

Parameter	(data pts)	95	45	22	14	13	12
ED_{50}		0.50	0.49	0.49	0.49	0.47	0.49
Upper FL	0.60	0.62	0.65	0.76	0.76	none	
				(.64)	(.72)	(.72)	(.63@85%)
Lower FL		0.41	0.37	0.36	0.20	0.002	none
				(.36)	(.24)	(.10)	(.34@85%)
Slope of Probit		4.0	5.1	10.2	8.7	8.0	7.8
Ratio(ED_{84} / ED_{50})		1.81	1.57	1.27	1.28	1.36	1.35
ED_{99}		1.91	1.40	0.84	0.90	0.93	0.97
ED_{01}		0.13	0.17	0.29	0.26	0.24	0.24
G-o-F test Value		2.13	1.46	0.12	0.73	1.17	0.48
Prob>Chi-Sq		0.98	0.98	0.99	0.98	0.95	0.99

Analysis of Table 7

From this data set, it appears that the minimum number of data points to have reasonable FLs for this distribution would be about 22 data points because even by doubling or quadrupling this number, the FLs change minimally. With only 12 data points, SAS could not calculate the FLs, and the limits were extremely wide with only 13 points. With 14 points, however, the FLs were still not reasonable, but were very close to the limits. Again, there are slight differences between the fiducial limits at the lower number of data points which can be attributed to the different methods in calculating the fiducial limits.

Next, data points were added to the 14-point set in an attempt to reduce the spread in the FLs to equal those for the 95-point data set. Table 8 gives the results of adding 8 data points in various combinations to the 14-point data set in Table 7 above. The desire was to minimize the total number of exposures or samples and yet obtain the large sampling results as those obtained for 95 data points. The starting data was as follows:

dose	0.2	0.3	0.4	0.5	0.6	0.7	0.8
shots	1	2	3	2	3	2	1
# yes	0	0	1	1	2	2	1

Table 8. Additional Data Points Added to Optimize the FLs and Minimize the Number of Trials. (Add eight data points: four data points added at ED_{25} and four data points added at ED_{75} or at ED_{08} and at ED_{92}.)

Data pts added	ED_{50}	Upper FL	Lower FL	Slope	G-of-F	Chi-Sq
Original 95 pts	0.50	0.60	0.41	4.0	2.13	0.98
Original 22 pts	0.49	0.65	0.36	10.2	0.12	0.99
Original 14 pts	0.49	0.76	0.20	8.7	0.73	0.98
4 - 0.5 @ "1"	0.49	0.69	0.27	8.7	0.75	0.98
4 - 0.5 @ "0"						
4 - 0.4 @75% -"1"	0.48	none	none	4.0	4.16	0.52
4 - 0.6 @25% -"1"		(0.71@85%)	(0.31@85%)			
4 - 0.4 @50% -"1"	0.48	0.79	0.25	5.7	1.83	0.87
4 - 0.6 @50% -"1"						
4 - 0.4 @25% -"1"	0.49	0.62	0.36	8.3	0.57	0.99
4 - 0.6 @75% -"1"						
4 - 0.4 @ 0% -"1"	0.49	0.58	0.40	12.7	0.09	1.00
4 - 0.6 @100%-"1"						
4 - 0.3 @25% -"1"	0.47	0.67	0.32	5.5	0.24	1.00
4 - 0.7 @75% -"1"						
4 - 0.3 @ 0% -"1"	0.48	0.59	0.36	10.5	1.44	0.92
4 - 0.7 @100%-"1"						

Analysis of Table 8

With 95 data points, the FLs were 0.60 and 0.41 as compared to the 0.76 and 0.20 with only 14 data points. When all eight data points were added at the ED_{50} level, four "1"s and four "0"s, the FLs changed very little. Next, four data points were added at the ED_{25} level and four data points added at the ED_{75} level in various combinations as shown in the table. With three "1"s and a zero added at 0.4 level, and three "0"s and a "1" added at 0.6 level, the FLs could not be calculated with SAS because the probit curve had predicted the reverse probabilities (i.e., three "0"s and a "1" at 0.4, and the reverse at 0.6). EZ-Probit was able to calculate the FLs at the 85% confidence level, and the limits are shown in parentheses. It should be pointed out that the Prob>Chi-Sq was only 0.52, which showed very poor correlation. Next, two "1"s and two "0"s were added at each point, which was still above the predicted values and, therefore, the FLs were not helped. Also, the Prob>Chi-Sq was still less than 0.90, which indicated that more data would be needed before being acceptable. Whenever the predicted results were added (i.e., three "0"s and one "1" added at 0.4, and three "1"s and one "0" added at 0.6, the FLs were drawn closer together and the Prob became 0.99). When four "0"s were added to 0.4 and four "1"s were added to 0.6 levels, the FLs were actually closer that those obtained with

95 data points. Other data points in two different combinations were added to the ED_{08} (level=0.3) and an ED_{92} (level=0.7) predicted levels as shown in the table. In both cases, the FLs were brought closer together but not as much as before.

CAUTION: It must be pointed out that the additional data added to the above table was always added with equal number of data points above and below the ED_{50} values at dosage level exactly spaced above and below the ED_{50} value. This symmetry of adding data is necessary to prevent having a biased estimate of the ED_{50}. Thus, data may not be arbitrarily added to an existing data set without carefully designing the additional experimental data sets.

Another data set which is encountered sometimes with real data is the step function response in which the responses (0 or 1) are all "0" below a certain level or dose and all "1" above the level or dose. This data set cannot be analyzed graphically or by the Kärber method because the slope of the probit curve would be a vertical straight line, and no points would fall on the graph. Also included was an increasing number of points overlapping from 1 and 2 points to a constant or flat response of 50% between two levels. In Table 9, the response of p = .5 is constant for levels of 0.4, 0.5, and 0.6 in the 4th column. Columns 5 and 6 both have p = .5 for levels .4, .5, and .6 and, in addition, levels .3 and .7 have either 3 out of 10 shots or 5 out of 10 shots for p = .3 or p = .5. The data in column 5 with p =.3 for level 0.3 is 30% (i.e., three "1"s out of 10 trials and three "0"s in level .7, while in the 5th column at p = .5, the response is 50% for both the 0.3 and 0.7 levels). In other words, the flat response occurs between 0.3 and 0.7 levels or for 5 levels in column 6. All computer runs with SAS contained 100 data points.

Table 9. SAS Probit Calculations for Step Functions and Constant Response Levels. Step distribution at 0.5 and p = .5 at .3, .4, .5, .6, and .7. One hundred data points used. SAS Probit and EZ-Probit gave identical results as shown.

Parameter	Step function no overlap	1 pt overlap	2 pts overlap	p=.5 (.4,.5,.6)	p=.3 (.3,.7)	p=.5 (.3,.7)
ED_{50}	0.5	0.51	0.49	0.48	0.46	0.44
Upper FL	none	0.55	0.54	0.53	0.53	0.59
Lower FL	none	0.46	0.45	0.42	0.38	0.28
Slope of Probit	89.8	20.6	15.4	8.2	5.2	4.2
Ratio(ED_{84} / ED_{50})	1.02	1.12	1.18	1.33	1.56	1.7
ED_{99}	0.53	0.66	0.70	0.92	1.28	1.60
ED_{01}	0.47	0.39	0.35	0.25	0.16	0.12
G-o-F Value	0.00	0.47	0.21	10.1	8.46	16.2
Prob>Chi-Sq	1.00	1.00	1.00	0.26	0.39	0.04

Analysis of Table 9

With no overlapping of data points, neither SAS Probit nor EZ-Probit could calculate FLs, and the calculated slopes from both were very large. Also note that the ED_{99} and ED_{01} are almost the same as the ED_{50}. As the number of overlapping data points increases and the slope of the probit decreases very rapidly, there is not much change in the fiducial limits. For the last two columns, adding "1"s to the 0.3 level decreased the ED_{50} value and increased the spread between the FLs. This occurs because the number of "1"s and "0"s added did not match predicted values for the levels .3 and .7. Also note that in the last two columns the value was greater than 10 and the Prob was between 0.39 and 0.04. These are very different values from those calculated for a normal distribution and would indicate that more data points need to be taken even though the FLs may appear to be reasonable.

CONCLUSIONS

The main conclusion to be drawn from this study is that the exact probit procedure is the one that should be used on any set of data and that the graphical solution is not adequate with todays computer technology. As quoted by Frisch[6], "Graphical solutions very often are adequate but complications may make this type of solution unsatisfactory -- and-- the exact probit method of solution is the most advantageous since it conveys the maximum amount of information from the data". Thus with the computer solutions available today, it is unthinkable to resort to graphical or calculator solutions such as the Kärber method for only a first approximation.

Many other conclusions may be drawn from these data sets, but the main consideration is the ability to maximize the reliability of the experimental procedure with a minimum number of data points. It was shown that the ED_{50} may be obtained with FLs at their 95% confidence interval with as few as 13 data points, even though the span between the FLs may be wide. It was also shown that with only eight more data points added, the FLs could be as good as or better than if 95 data points had been taken, depending on the data points added. It may be concluded that when data points are added to an existing data set with 95% FLs, whenever those data points are at doses below the ED_{50} values and the percentages of yes or "1" is less than the ED percentage at that dose, the combined data set will be helped. If the percentages of yes or "1" are greater than the ED value, the data set will not be helped. In other words, for a fixed number of data points added below the ED_{50}, the percentage of positive responses must be less than the ED percent at that dose or the lower FL will be further from the ED_{50} value and may not be calculable at the 95% confidence level. The reverse is true for data points added above the ED_{50} where the percentages of yes or "1" must be greater than the ED percentage at that dose level. The reasoning for the above is obvious due to the maximum likelihood estimators of the probit procedure, and when the new data does not match the probit probabilities, the combined data set becomes either better or worse.

Another conclusion with the most significance is that it is relatively easy to write your own computer program to calculate the probability curve and the FLs for any level of significance desired. EZ-Probit has been shown to be equivalent to the SAS Probit with the exception of the fiducial limits for a limit number of data points, and also has the added flexibility to be able to determine the FLs at any desired level of confidence. EZ-Probit is a lot easier to use, faster than SAS Probit, and has more capability. Since EZ-Probit has been installed on the P-drive, it is available for use by anyone, anywhere, anytime.

REFERENCES

1. Kärber, G. Arch Exp. Path. u. Pharmakol. 162: (1931).

2. DeBeer, B.J. Calculation of biological assay results by graphic methods. J. of Pharm. & Exp. Ther. 85:1 (1945).

3. Finney, D.J. Probit Analysis. 3rd edition, Cambridge Univ. Press (1971).

4. Foster, D.H. and Bischof, W.F. Thresholds from psychometric functions: Superiority of bootstrap to incremental and probit variance estimators. Psychological Bull. 109 (1): 152-159 (1991).

5. SAS Institute, Inc. SAS Campus Drive, Cary, NC 27513.

6. Frisch, G. D. Quantal Response Analysis. Memo. Report M70-27-1 of Joint AMRDC-AMC Laser Safety Team, US Army, Frankfort Arsenal, Philadelphia, PA 19137 (1970).

7. Cain, C.P., Toth, C.A., DiCarlo, C.D., Stein, C. D., Noojin, G.D., Stolarski, D.J., and Roach, W.P., Visible retinal lesions from ultrashort laser pulses in the primate eye, Invest. Ophthalmol. Vis. Sci 36(5):879-888 (1995).

8. SAS/STAT Software Syntax Ver. 6, 1st Ed. SAS Institute, Inc., Cary, NC 27513.

APPENDIX A:

PC COMPUTER PROGRAM FOR SAS PROBIT

The following program was written for SAS to operate on the PC computer and prints the outputs. A brief explanation of what the code does follows the program. A complete description of the terminology is given in the SAS/STAT Software Syntax manual.[7]

```
1.      filename testdata 'filename.dat';
2.      title 'filename.dat';
3.      data indata;
4.            infile testdata;
5.            input energy mvl;
6.      run;
7.
8.      proc sort data = indata;
9.            by energy;
10.     run;
11.
12.     proc print data = indata;
13.
14.     run;
15.
16.     proc sort data = indata;
17.           by desending energy;
18.     run;
19.
20.     proc probit data = indata outest = mvldata order = data covout Hprob = 0.10
        lackfit log10 inversecl;
21.          class mvl;
22.          model mvl = energy / lackfit d = normal corrb covb inversecl;
23.          output out =  mdat prob = p xbeta = xB std = sd;
24.     run;
25.
26.     goptions target=winprtg;     /* grey-scale postscript */
27.
28.     goptions rotate = landscape;
29.
30.     proc gplot;
31.           axis logbase = 10 logstyle = power;
32.           plot mvl*energy p*energy  / overlay;
33.     run;
```

The first 6 lines read the data from a file into a SAS data set variable named "indata." The file should be a two-column list of numbers separated by a space with each line terminated with a return character. The first column should be the dose or energy delivered and will be read into the "energy" array. The second column should be the yes/no response (1 = yes, 0 = no) to the dose. This column will be read into the "MVL" array. Lines 8 through 10 sort the data set by energy in ascending order for printout in lines 12 through 14. Lines 16 through 18 sort the data again by energy in descending order for the probit procedure. This last sort is to ensure that the first data point is a yes response. The SAS probit procedure assumes that the first data point's response is the value that will be considered a "yes" response for the current run. The call to the probit procedure begins on line 20 and ends on line 24. The following is a description of the flags used in the call to the probit procedure and their purpose.

data	Supplies the name of the SAS data set to be used by the probit procedure.
outest	Supplies the name of the SAS data set where the output will be written.
covout	Writes the covariance estimates to the **outest** data set.
Hprob	Specifies the probability level to indicate the goodness of fit.
lackfit	Performs a Pearson chi-square test of the fit and log-likelihood ratio chi-square test of the fit.
log10	Converts the independent variable to its log base 10 value for analysis.
inversecl	Computes the confidence limits for the independent variable

The "class" statement on the next line names the response variable (mvl) for the probit analysis. The "model" statement specifies the response variable and the independent variable it is based on. The "model" statement also has several flags associated with it that specify output options and probit parameters. The following is a list of the options specified and their purpose.

lackfit	Performs a Pearson chi-square test of the fit and log-likelihood ratio chi-square test of the fit.
d	Specifies the type of statistical distribution to use.
corrb	Prints the estimated correlation matrix of the parameter estimates.
covb	Prints the estimated covariance matrix of the parameter estimates.
inversecl	Computes the confidence limits for the independent variable.

The last statement is the "output" statement. This statement associates data output from the probit procedure with variable names that can be used later in the SAS run. The rest of the statements in the script specify printing options for the graph output.

APPENDIX B:

EZ-PROBIT COMPUTER PROGRAM

The following program was written by Captain Lonnie Manning to perform the probit calculations as outlined in Finney's book.

EZ-PROBIT

TOPIC: Overview of how to calculate ED_{50} using the EZ-Probit method.

Starting point for this type of "statistic" is the probability function

$$P(x) = \int_{-\infty}^{t=x} f(t)\,dt$$

EZ- Probit assumes a normal distribution for this type of data. Thus,

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(\frac{-(x-\mu)^2}{2\sigma}\right).$$

Making the (probit) transformations

$$Y-5 = \frac{(x-\mu)}{\sigma} \text{ and } t = \frac{(x-\mu)}{\sigma},$$

we get

$$P(Y-5) = \int_{-\infty}^{t=Y-5} \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{t^2}{2}\right)dt$$

The normal equivalent deviate (NED) uses, $Y = \frac{(x-\mu)}{\sigma}$, which can take on negative values. Y-5 typically does not.

THE CALCULATION

THE DATA

x_i	Log_{10} of the i^{th} unique energy level.
n_i	The number of trials at x_i
r_i	The number of lesions for x_i

$p_i = \frac{r_i}{n_i}$ Estimated $P(x)$.

WORKING PROBITS

The equation, $Y - 5 = \frac{x - \mu}{\sigma}$, can be rewritten as:

$$Y_i = a \cdot x_i + b .$$

The basic concept behind probit is to find the slope and intercept of the above equation. Straight-forward linear regression (LR) or weighted LR does not provide the most accurate description of this system. This is accomplished by introducing "working probits" (WP).

The WP is derived from the maximum likelihood equation and is given by:

$$y_i = Y_i + \frac{(p_i - P(Y_i - 5))}{z_i} ,$$

where

$$z_i = \frac{1}{\sqrt{2\pi}} \exp(-\frac{(Y_i - 5)^2}{2}) .$$

For $p_i = 0$, we have the famous y_0

$$y_0 = Y_i - \frac{P(Y_i - 5)}{z_i} .$$

For $p_i = 1$, we have the famous y_{100}

$$y_{100} = Y_i + \frac{1 - P(Y_i - 5)}{z_i} = Y_i + \frac{Q(Y_i - 5)}{z_i}$$

PERFORMING THE CALCULATIONS

Step 1.

Choose an approximate slope and intercept for the line, $Y_i = ax_i + b$.
For most cases, $Y_i = 5$, will do nicely as a starting point.

Step 2.

Calculate a new slope, a' and intercept b' using weighted LR with a weighting factor of:

$$w_i = \frac{z_i^2}{\mathrm{P}(Y_i - 5)(1 - \mathrm{P}(Y_i - 5))} .$$

Step 3.

Calculate new y_i using the results of step 2.

$$Y_i' = a'x_i + b'..$$

Step 4.

Do steps 2 and 3 until it converges.

Step 5.

Once the slope and intercept are obtained, Log (ED_{50}) can be calculated.

Since $P(Y-5) = \int_{-\infty}^{t=Y-5} \frac{1}{\sqrt{2\pi}} \exp(-\frac{t^2}{2})dt = \frac{1}{2}$, for Y=5,

Log(ED50) = $x_{50} = \frac{5-b'}{a'}$, or $x_{50} = \frac{5-\bar{y}+a'\cdot\bar{x}}{a'}$.

Note: In reality, one calculates the equation, $Y = \bar{y} + a'\cdot(x-\bar{x})$, since $\bar{y}$, a' and $\bar{x}$ are used later to calculate FLs.

THE FOLLOWING PROGRAM PERFORMS ALL OF THE REQUIRED CALCULATIONS:

Subroutine 1. EZ-IO.H

```
int iofile (long double *a,int *b,string f_name,int lin_log){

long double DUM_a ,en;

ifstream inf(f_name.c_str(), ios::in | ios::nocreate );

int i,nmax,k,DUM_b,low,hit;

        if (lin_log==1)
        {
         for (i=0;inf.eof()!=1;i++)
         {
               inf>>en;
               inf>>hit;
                if (inf.eof()!=1)
                      {     a[i]=log10l(en);
                               b[i]=hit;
                      }
        ;};

}
```

```
          else
          {
                    for (i=0;inf.eof()!=1;i++)
                    {
                     inf>>en;
                     inf>>hit;
                              if (inf.eof()!=1)
                              { a[i]=en;
                                        b[i]=hit;
                              }
                    ;};
                                        inf.close();
          }
                                        nmax=i-2;

for (k=0;k<=nmax;k++)             //Cheap & dirty sorting procedure
{
          low=k;

          for (i=k+1;i<=nmax;i++)
          { if(a[i]<=a[low]) low= i; };

          DUM_a=a[low];DUM_b=b[low];
          a[low]=a[k];b[low]=b[k];
          a[k]=DUM_a;b[k]=DUM_b;
};
return nmax;
};
```

Subroutine 2. N-INV.H

```
routine 2. //The Inverse of the standardized Normal distribution use for fiducial limits
int set_ninv(long double * );
int set_ninv(long double *a ){
a[0]= -2.3263478740408411;      //norm inverse of .01
a[1]= -2.05374891063182305;
a[2]= -1.88079360815125094;     //norm inverse of .03
a[3]= -1.75068607125216998;
a[4]= -1.64485362695147271;     //norm inverse of .05
a[5]= -1.55477359459685354;
a[6]= -1.47579102817917073;
a[7]= -1.40507156030963255;
a[8]= -1.34075503369021637;
a[9]= -1.28155156554460047;   //norm inverse of .10
a[10]= -1.03643338949378955;  //norm inverse of .15
a[11]= -0.841621233572914189;
a[12]= -0.674489750196081728;
a[13]= -0.524400512708040724;
a[14]= -0.385320466407567563;
a[15]= -0.253347103135799736;
a[16]= -0.125661346855073969;
a[17]= 0.00000;                 //norm inverse of .5
a[18]= 0.125661346855074109;
a[19]= 0.25334710313579988;
a[20]= 0.385320466407567713;
a[21]= 0.524400512708040884;        //by.05
a[22]= 0.674489750196081877;
a[23]= 0.84162123357291437;
a[24]= 1.03643338949378979;
a[25]= 1.28155156554460076;  //norm inverse of .90
a[26]= 1.34075503369021653;
a[27]= 1.40507156030963272;    //by .01
a[28]= 1.47579102817917092;
a[29]= 1.55477359459685374;
a[30]= 1.64485362695147295;
a[31]= 1.75068607125217026;
a[32]= 1.8807936081512513;
a[33]= 2.05374891063182355;
a[34]= 2.32634787404084201;
return 0;}    ;
```

Subroutine 3. Probit.H

```
#include <iostream.h>
#include <math.h>
#include <stdio.h>

#define PI 3.14159265358979323846264338327950

//The natural log of the gamma function

long double gamln(long double n)
{long double sum;
 if(n==0.5) return logl(sqrtl(PI));
if (n==1.0) return logl(1.0);
if (n==2.) return logl(1.0);
sum=0;
if((n-(int)n)!= 0.5)
{ for(int i=0;i<=n-2.;i++)
        {
        sum=sum+logl(n-1.0-i*1.0);
        }
}
else
{
        sum=logl(sqrtl(PI));
        for(int i=0;i<=(int)(n-1.0);i++)
  sum=sum+logl(n-1.0-i*1.0);
}
return sum;
}
//____________________________________________________________

// The normal distribution -- series approximation

long double
P_ser(long double x){
```

```
long double af,an,sum,px,dj;
af=0.;an=0.;sum=0.;px=0.;dj=0.;
int j,m_x;
m_x=1;
        if (x<0.0) {x=-x;m_x=-1;}
an= 1.;
sum=an;
for(j=1;j<=500;j++)
{
        af=sum;
        dj=j*1.;
        an=-an*x*x*(2.0*dj-1.)/(dj*2.0+1.0)/(2.*dj);
        sum=sum+an;
        if ((sum-af)==1.e-18) break;};  /* get out*/

sum=x*sum/sqrtl(2.*PI) ;

/*cout.precision(22);*/
px=1./2.+m_x*sum;
return px;
}

//Factorial function -- only good for integer and half-integer values

long double
fac(long double n){
if(n==-1.) return sqrt(-1);
if (n==-.5) return sqrtl(PI);
if (n==0.) return 1.;
if (n==1.) return 1.;
return n*fac(n-1.);
}

//The Gamma function

long double
gam(long double x){return fac((x-1.0));}

//
long double
gser(long double a,long double x)
{
        long double an,sum,eps;
        an=1.0;
        eps=1.e-20;
        sum=an;
                for(int n=1;n<=5000;n++)
                        {
                                an=x*an/(a+n*1.0);
                                sum=sum+an;
                                if(fabsl(an)<fabsl(sum)*eps) break;
                        }
return sum*expl(-x)*powl(x,a)/a;
}
```

```
//_______________________________________________________________

//calculates  p(chi,v) using the continued fraction method

long double
p_chi(long double CHI_2,int df)
{
          long double aj,bj,cj,dj,fj,tiny,x,delta_j,eps,a,res;
          int j;
          tiny=1.0e-45;
          eps=1.0e-20;
          a=df/2.0;
          x=CHI_2/2.0;
                    if (x<a+1.0){res= 1.-gser(a,x)/gam(a);goto get_out;};
          bj=x+1.0-a;
          if(bj==0.0) bj=tiny;
          cj=1.0/tiny;
          dj=1.0/bj;
          fj=dj;
          for(j=1;j<=5000;j++)
          {
                    aj=-j*(j-a);
                    bj+=2.0;
                    dj=aj*dj+bj;
                    if (fabsl(dj)<tiny) dj=tiny;
                    cj=bj+aj/cj;
                    if (fabsl(cj)<tiny) cj=tiny;
                    dj=1.0/dj;
                    delta_j=cj*dj ;
                    fj=delta_j*fj;
                    if(fabsl(delta_j-1.0)<eps) break;
          }
          res=powl(x,a);
          res=res*expl(-x);
//if(res<eps) return 0.0;
          res=res/gam(a);
          res=res*fj;
get_out:
           if (res>1.0) return 1.0;
          return res;
}
//--------------------------------------------------------------------
//Probability -Inf to x for the Normal distribution

long double
P(long double x)
{
 long double Qx;
 if (fabsl(x)<=.6) return P_ser(x);
 Qx= p_chi(x*x,1.)/2.;
 if (x<0) return Qx;
return 1.-Qx;

}
```

```
//-------------------------------------------------------------------------------
// The inverse of the Probability for the Normal distribution

long double
norms_inv( long double prob_x )
{
          long double d,a0,a1,Za0;
          int i;
          a0=.1;
          for (i=1; i<=2000;i++)
          {
                    Za0=expl(-a0*a0/2.0)/sqrtl(2.0*PI);
                    a1=a0-(P(a0)-prob_x)/Za0;
                    d=a1-a0;
                    if (fabsl(d)<=1.e-18) break;
                    a0=a1;
          };
//cout << "d ="<<d<<"  a1= "<<a1<<endl;
return a1;
}
//-------------------------------------------------------------------------
// The contnued fraction appraoximation of the Incomplete Bets Function

long double
betacf(long double a,long double b,long double x)
{
          int m,m2;
          long double eps,FPMIN,aa,c,d,del,h,qab,qam,qap;
          eps=1.e-20;
          FPMIN=1.e-45;
          qab=a+b;
          qap=a+1.0;
          qam=a-1.0;
          c=1.0;
          d=1.0-qab*x/qap;
          if(fabsl(d)<FPMIN) d=FPMIN;
          d=1.0/d;
          h=d;
          for (m=1;m<=5000;m++)
          {
                    m2=2*m;
                    aa=m*(b-m)*x/((qam+m2)*(a+m2));
                    d=1.0+aa*d;
                     if(fabsl(d)<FPMIN) d=FPMIN;
                    c=1.0+aa/c;
                     if(fabsl(c)<FPMIN) c=FPMIN;
                    d=1.0/d;
                    h*=d*c;
                    aa=-(a+m)*(qab+m)*x/((a+m2)*(qap+m2));
                    d=1.0+aa*d;
                     if(fabsl(d)<FPMIN) d=FPMIN;
                    c=1.0+aa/c;
                     if(fabsl(c)<FPMIN) c=FPMIN;
                    d=1.0/d;
                    del=d*c;
                    h*=del;
            if (fabsl(del-1.0)<eps) break;
            }
return h;
}

// The incomplete Beta function

long double
betai(long double a, long double b,long double x)
{
            long double bt;

            if(x<0.0||x>1.0) sqrtl(-1);
            if(x==0.0 || x==1.0) bt=0.0;
            else
            bt=expl(gamln(a+b)-gamln(a)-gamln(b)+a*logl(x)+b*logl(1.0-x));

            if(x< (a+1.0)/(a+b+2.0))
                        return bt*betacf(a,b,x)/a;
            else
                        return 1.0-bt*betacf(b,a,1.0-x)/b;
}
 // Prob for the Students t distribution
long double
student_t(long double t,long double v/*deg of freedom*/)
{
            return 1.0-betai(v/2.,1./2.,v/(v+t*t));
}

//-----------------------------------------------------------------------
// The inverse of Student's t distribution

long double
student_t_inv( long double prob_t,long double v )
{
            long double d,a0,a1,A0,A00;
            int i;
            a0=0;
A00=(0.5*logl(v)+gamln(0.5)+gamln(v/2.0)-gamln((1.0+v)/2.0));
//+((v+1.0)/2.0)*logl(1.0+a0*a0/v));

            for (i=1; i<=500;i++)
            {
A0=A00+((v+1.0)/2.0)*logl(1.0+a0*a0/v);
                        a1=a0-(student_t(a0,v)-prob_t)*expl(A0)/2.0;
                        if (fabsl(a1-a0)<=1.e-18) break;
                        a0=a1;
            };
//cout << "d ="<<d<<"  a1= "<<a1<<endl;
return a1;
}
```

Main Program, Probit.CPP

```
#include <owl\applicat.h>
#include <owl\framewin.h>
#include <owl\dialog.h>
#include <owl\dc.h>
#include <owl\opensave.h>
#include <owl\edit.h>
#include <owl\checkbox.h>
#include <owl\validate.h>
#include <c:\bc4\include\owl\window.rh>
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <math.h>

#pragma hdrstop
#include "c:\bc4\mine\ez_io.h"
#include "c:\bc4\mine\pobit.h"
#include "c:\bc4\mine\n_inv.h"


#define CM_EMPINPUT 201
#define CM_FILEOPEN      0x100
#define CM_COLOR         0x101

#define CM_HELPABOUT     0x200

long double PowL(long double xx,int ln_lg){if (ln_lg==1) return powl(10.0,xx);return xx;};
long double Z(long double YY){
return expl(-YY*YY/2.)/sqrtl(M_PI*2.);};
char *fn;
char  fln[24];

#define MAXNAMELEN  35
#define MAXSSLEN    12
#define MAXIDLEN    4
#define MAXDEPTLEN  7
#define MAXSECLEN   3

struct TParamStruct {
char DefaultDir[50];
char Converg[MAXIDLEN];
 char OprecEdit[3];
char Confidence[7];
 char PerCent[3];
  BOOL Log_Dose;
  BOOL PermFile;
  BOOL Exempt;
};

//---------------------------------------------------------------------------

class TParamDlg : public TDialog {
 public:
        TParamDlg(TWindow* parent, const char* name, TParamStruct& tx);
};

TParamDlg::TParamDlg(TWindow* parent, const char* name, TParamStruct& tx)
 : TDialog(parent, name),
        TWindow(parent)
{           TEdit *edit;

edit=new TEdit(this, 101, sizeof(tx.DefaultDir));
//edit->SetValidator(new TFilterValidator("A-Za-z. \\\: "));
edit = new TEdit(this, 103, sizeof(tx.Converg));
        edit->SetValidator(new TRangeValidator(12, 18));
edit = new TEdit(this, 104, sizeof(tx.OprecEdit));
        edit->SetValidator(new TRangeValidator(0, 21));
edit = new TEdit(this, 105, sizeof(tx.Confidence));
        edit->SetValidator(new TPXPictureValidator("#.####"));
        edit = new TEdit(this, 102, sizeof(tx.PerCent));
        edit->SetValidator(new TRangeValidator(0, 100));
new TCheckBox(this, 106);
new TCheckBox(this, 107);
new TCheckBox(this, 109);
//              TEdit* edit;

 /* edit = new TEdit(this, 104, sizeof(tx.DeptEdit));
edit->SetValidator(new TPXPictureValidator("Sales,Dev,Mfg")); /*
 edit = new TEdit(this, 105, sizeof(tx.SecEdit));
edit->SetValidator(new TPXPictureValidator("11,12,13,14,15"));
        */

 TransferBuffer=(void far *)&tx;

}

class TCommDlgWnd : public TFrameWindow {
 public:
        TCommDlgWnd(TWindow*, const char* );
        ~TCommDlgWnd();

        void    Paint(TDC&, BOOL, TRect&);
        void    CmFileOpen();
        void    CmHelpAbout();
        void    CmEmpInput();
        TColor    Color;
        TFont*    Font;

         TOpenSaveDialog::TData FilenameData;

void EvSize(UINT sizeType, TSize& size);

 private:
        TParamStruct ParamStruct;

 DECLARE_RESPONSE_TABLE(TCommDlgWnd);
};
```

```
DEFINE_RESPONSE_TABLE1(TCommDlgWnd, TFrameWindow)

          EV_WM_SIZE,
  EV_COMMAND(CM_FILEOPEN, CmFileOpen),
          EV_COMMAND(CM_EMPINPUT, CmEmpInput),
  EV_COMMAND(CM_HELPABOUT, CmHelpAbout),
END_RESPONSE_TABLE;


TCommDlgWnd::TCommDlgWnd(TWindow* parent, const char* title)
  : TFrameWindow(parent, title),
          TWindow(parent, title),
          FilenameData(OFN_FILEMUSTEXIST|OFN_HIDEREADONLY|OFN_PATHMUSTEXIST,
                                        "Data Files (*.dat)|*.dat|All Files (*.*)|*.*|",
                                        0, "", "*")
{
          memset(&ParamStruct, 0, sizeof ParamStruct);

          ParamStruct.Log_Dose=TRUE;
          ParamStruct.PermFile=FALSE;
          strcpy(ParamStruct.Converg,"18");
          strcpy(ParamStruct.DefaultDir,"C:\\probit\\");
                    strcpy(ParamStruct.OprecEdit,"6");
                    strcpy(ParamStruct.Confidence,"0.1000");
                              strcpy(ParamStruct.PerCent,"95");
  AssignMenu("CMDLGAPMENU");
                    // Set up the menu
          Color = TColor::Black;        // Use black as the default color
  Font = 0;
                                                                  // Empty the handle to the font
}

TCommDlgWnd::~TCommDlgWnd()
{
  delete Font;
}
//
// We need to invalidate the entire area, not just the clip area so that
// paint gets called correctly
//
void
TCommDlgWnd::EvSize(UINT sizeType, TSize& size)
{
          Invalidate();
          TFrameWindow::EvSize(sizeType,size);
}

//
// Display the file name using the selected font in the selected color.
//
void
TCommDlgWnd::Paint(TDC& paintDC, BOOL, TRect&)
{
  paintDC.SetTextColor(Color);
```

```
 paintDC.SetBkColor(::GetSysColor(COLOR_WINDOW));
 if (Font)
         paintDC.SelectObject(*Font);
 paintDC.DrawText(fn, strlen(fn),
                                                       GetClientRect(), DT_CENTER |
DT_WORDBREAK);
}

//
//
void
TCommDlgWnd::CmFileOpen()
{
char  msg[50];
//MessageBox(itoa(ParamStruct.Log_Dose,msg,10),"ASDASD",MB_OK);

strcpy(msg,"                              ");
fn=new char [512];
          if (TFileOpenDialog(this, FilenameData).Execute() == IDOK)
 {
         strcpy(msg,FilenameData.FileName);
 FlashWindow(1);
 SetCursor(NULL,IDC_WAIT);

 int  lin_log, PermData;
 lin_log=ParamStruct.Log_Dose;
 long double confidence,tolerance,eps,outprec,percent;
 outprec=atoi(ParamStruct.OprecEdit);
         eps=pow10l(-atoi(ParamStruct.Converg));
         confidence=_atold(ParamStruct.Confidence);
         if (confidence>1.0) confidence=1.0;
         if (confidence <0) confidence =0;
         percent=atoi(ParamStruct.PerCent)*1.0/100.00;  //outprec=15;
         PermData=ParamStruct.PermFile;
         //change to default to 6 for clarence
          long double * en;      en= new long double [200];
          long double * len;     len= new long double [200];
          long double * y;       y= new long double[200];
 //    long double * Y;             Y= new long double [200];
          long double * p;       p= new long double [200];
          long double * ninv;    ninv= new long double [35];
          int * hit;             hit= new int  [200];
          int * n;               n= new int  [200];
          int * H;               H= new int  [200];
          int * r;               r= new int  [200];
          long double * w;       w= new long double [200];
          long double YY,cr,s0,s1,s3,S1,S2,S3,Sxx,Sxy,Syy,z,nw,x_,y_,chi_2xy,chi_2LR,g,FL,D,intercept;
          long double foo,oof, slope,Yint, EXP_P,yy, log_ED50, ED50,m,dum,t,h,FLU,FLL;
          int j,k,kmax,nmax,bigslope,ntot,newt;
                                                   newt=0;
           bigslope=0;
          string filename,ofile,ofile2,ofile3,pnt,pth,hdr1,hdr2;
hdr1=" Prob      Log[Dose]       Log[Lower Limt]    Log[Upper Limit]    \n";
hdr2=" Prob        Dose           Lower Limit        Upper Limit        \n";
```

```
            set_ninv(ninv);
            pnt="";
            filename=msg;
            strcpy(fn,filename.c_str());
             for (k=0;k<=filename.rfind("\\");k++) pth= pth+filename[k];
             strcpy(ParamStruct.DefaultDir,pth.c_str());
            for (k=0;k<=filename.rfind(".");k++) pnt= pnt+filename[k];
                    if(PermData==FALSE) pnt="tmp.";
            ofile=pnt+"out";


           if(lin_log==1)
            { ofile2=pnt+"log";
                    ofile3=pnt+"grf";

            }
                    else
            { hdr1=hdr2;
                    ofile2=pnt+"grf";
            }

            ofstream  otf( ofile.c_str(), ios::out );
            ofstream otf2( ofile2.c_str(), ios::out );
            ofstream otf3( ofile3.c_str(), ios::out );
            otf.precision(outprec);
 //    otf<<ofile <<'\n';
                    otf.precision(outprec);
            nmax=iofile(en,hit,filename,lin_log);


/*the following procedure acts like a "pivot table"*/
r[0]=hit[0]; H[0]=0; k=1; foo=en[0]; ntot=hit[0];
           for (j=1;j<=nmax;j++)
           {  ntot=ntot+hit[j];
                    if(en[j]==foo)
                            r[k-1]=hit[j]+r[k-1];
                    else                /* new energy*/
            {
                    H[k]=j;
                    r[k]=hit[j];
                    k++;
                    foo=en[j];
            }
           }

kmax=k-1;
H[kmax+1]=nmax+1;  /*need to make sure H[kmax+1] exists*/
           for  (k=0;k<=kmax;k++)
           {
                    n[k]=H[k+1]-H[k];
                    j=H[k];
                    len[k]=en[j];
           }
//Initial values for slope and Intercept of the Probit
//for normit Yint ==0
```

```
slope=0.0;
Yint=5.0;
log_ED50=10.0;          //log_ED50=ED50 for lin_log=TAKE_LIN
dum=1.0;
t=norms_inv((long double)0.975);

do{
        oof=log_ED50;
        s0=0;s1=0;s3=0;S1=0;S2=0;S3=0,chi_2LR=0;
        for(k=0;k<=kmax;k++)
                {
                        p[k]=(1.0*r[k])/(1.0*n[k]);
                        yy=len[k]*slope +Yint;  //yy=expected probit
                        EXP_P=P(yy-5.);
                        YY=yy-5.0;
                        foo=p[k]-EXP_P;
                        if(foo==0.0)
                 { y[k]=yy;      //lim as YY->INF foo/z  =0, z=0
                         w[k]=0.0;
                 }
                         else
                 {
                         YY=YY*YY/2.0;
                         z=expl(YY) ;
                         z=z*sqrtl(M_PI*2.0);
                         z=1.0/z;
                         y[k]=yy+foo/z;
                         w[k]=z*z/(EXP_P-EXP_P*EXP_P);
                 }
                        nw=n[k]*w[k];
                        s1=s1+nw*len[k];
                        S1=S1+nw*len[k]*len[k];
                        s3=s3+nw*y[k];
                        S2=S2+nw*y[k]*len[k];
                        S3=S3+nw*y[k]*y[k];
                        s0=s0+nw;
                        //chi_2LR=chi_2LR+(r[k]*1.0- n[k]*EXP_P)*(r[k]*1.0-
n[k]*EXP_P)/n[k]/(EXP_P-EXP_P*EXP_P);
                };

                Sxx=S1-s1*s1/s0;
                Sxy=S2-s1*s3/s0;
                Syy=S3-s3*s3/s0;
                slope=Sxy/Sxx;
                x_=s1/s0;            //the new equation is Y=y_+slope*(x-x_);
                y_=s3/s0;
                Yint=y_-slope*x_;    //i.e.  x=0
                log_ED50=x_+(5.0-y_)/slope;
                if(slope>200.) {bigslope=1;break;}
         if(dum>55) break;           //Maximum number of iterations
                dum=dum+1.0;
                //tolerance = fabsl(oof-oof);
                tolerance = fabsl((log_ED50-oof));
} while (tolerance>=eps);
```

```
chi_2xy=Syy-Sxy*Sxy/Sxx;         //CHI Squared
m=log_ED50;                     //logED50 or ED50
if(chi_2xy<1.0e-18)             //Chi^2 = zero
{chi_2xy=0.0;oof=1.;}
else
oof=p_chi(chi_2xy,kmax-1);       //Probability of Chi^2, N-2 Degress ofFreedom
h=1.0;

if(fabsl(percent==0.95) )
 t=1.96;                        //t=student_t_inv(percent,INF);
else
t=norms_inv(1.-(1.-percent)/2.);  //Calculates t for new confidence level

if(oof<confidence)              //Check for good data
          {newt=1;
           h=chi_2xy/(kmax*1.0-1.0);     //heterogeneity factor
                    t=student_t_inv(percent,(long double)(kmax-1));};
intercept=y_ -slope*x_-5.0;
ED50=PowL(log_ED50,lin_log);
//cr=Sxy/sqrtl(Sxx*Syy) ;
//cr=cr*1.00;
oops:
if(bigslope!=1)                 //Fix for infinite slope
{
g=(t*t)*h/Sxx/slope/slope;
FL=m+g*(m-x_)/(1.0-g);
D= (1.-g)/s0+(m-x_)*(m-x_)/Sxx;
if(D<0.0){bigslope=1;goto oops;}
D=sqrtl(h)*t/slope/(1.0-g)*sqrtl(D);
FLL=FL-D;
FLU=FL+D;}
else
{h=0;
g=0;
FLL=m;
FLU=m;}
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++
otf<<"                    "<<filename.c_str()<<"\n\n"
          <<"\t\t   ONES = "<<ntot<<"\t\t ZEROES = "<<(nmax+1-ntot)<<'\n'
          <<"\t\t          h  = " <<h<<'\n'
          <<"\t\t          g  = " <<g<<'\n'
          <<"\t\t          t  = " <<t<<'\n'
          /*<<"\t\t          Snw = "<<s0<<'\n'
          <<"\t\t        Snwx = "<<s1<<'\n'
          <<"\t\t        Snwy = "<<s3<<'\n'
          <<"\t\t       Snwxx = "<<S1<<'\n'
          <<"\t\t       Snwxy = "<<S2<<'\n'
          <<"\t\t       Snwyy = "<<S3<<'\n'*/
          <<"\t\t         SYY = "<<Syy<<'\n'
          <<"\t\t         SXY = "<<Sxy<<'\n'
          <<"\t\t         SXX = "<<Sxx<<'\n';
          if(lin_log==1) otf<<"\t\t      LGED50 = "<<log_ED50<<'\n' ;
otf<<"\t\t          ED50 = "<<ED50<<'\n'
          <<"\t\t         FLU = "<<PowL(FLU,lin_log)<<'\n'
```

```
        <<'\t\t        FLL = "<<PowL(FLL,lin_log)<<"\n\n"
        <<" Pearson's CHI2 = "<<chi_2xy<<"\t DF = "<<(kmax-1)<<'\t'
        <<"    Prob>Chi-Sq = "<< oof<<"\n\n"
        <<'\t\t        XBAR = "<<x_<<'\n'        //only kmax-1=N-2 because
        <<'\t\t        YBAR = "<<y_<<'\n'        // k goes from 0 to kmax
        <<'\t\t    Intercept = "<<intercept<<'\n'
        <<'\t\t        Slope = "<<slope<<'\n'
        <<"Iteration = "<<(int) dum<<"\n\n\n\n"<<endl;
//-------------------------------------------------------------------------
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++
otf2<<"\n                    Input File "<<filename.c_str()<<"\n\n" ;
otf2<<hdr1  ;
//     +0.01  -0.881940268668475     -1.78261681790058      -0.572033298124713
otf2.precision(outprec);
otf2.setf(ios::showpos|ios::left);
otf2<<'\n';
                for(k=1;k<=10;k++)
                {     m=(ninv[k-1]-intercept)/slope;
                        FL=m+g*(m-x_)/(1.-g);
                        D=t/slope/(1.-g)*sqrtl((1.-g)/s0+(m-x_)*(m-x_)/Sxx);
                        otf2.width(5);
                        otf2<<(k*0.01)<<'\t';
                        otf2.width(18);
                        otf2<<m<<'\t';
                        if(bigslope!=1){
                        otf2.width(18);
                        otf2<<(FL-D)<<'\t';
                        otf2.width(18);
                        otf2<<(FL+D)<<'\n';}
                        else
                        {
                        otf2.width(18);
                        otf2<<" - "<<'\t';
                        otf2.width(18);
                        otf2<<" - "<<'\n';}
                };

                for(k=1;k<=16;k++)
                {     m=(ninv[k+9]-intercept)/slope;
                        FL=m+g*(m-x_)/(1.-g);
                        D=t/slope/(1.-g)*sqrtl((1.-g)/s0+(m-x_)*(m-x_)/Sxx);
                        otf2.width(5);
                        otf2<<(.1+k*0.05)<<'\t';
                        otf2.width(18);
                        otf2<<m<<'\t';
                if(bigslope!=1){
                        otf2.width(18);
                        otf2<<(FL-D)<<'\t';
                        otf2.width(18);
                        otf2<<(FL+D)<<'\n';}
                        else
                        {
                        otf2.width(18);
                        otf2<<" - "<<'\t';
```

```
                              otf2.width(18);
                              otf2<<" - "<<'\n';}
                    };

          for(k=1;k<=9;k++)
                    {     m=(ninv[k+25]-intercept)/slope;
                              FL=m+g*(m-x_)/(1.-g);
                              D=t/slope/(1.-g)*sqrtl((1.-g)/s0+(m-x_)*(m-x_)/Sxx);
                              otf2.width(5);
                              otf2<<(.9+k*0.01)<<'\t';
                              otf2.width(18);
                              otf2<<m<<'\t';
                              if(bigslope!=1){
                              otf2.width(18);
                              otf2<<(FL-D)<<'\t';
                              otf2.width(18);
                              otf2<<(FL+D)<<'\n';}
                              else
                              {
                              otf2.width(18);
                              otf2<<" - "<<'\t';
                              otf2.width(18);
                              otf2<<" - "<<'\n';}
                    };
//-------------------------------------------------------------------------------
if(lin_log==1) //do this part because we also want the file in regular non-log units
{
//++++++++++++++++++++++++Output tof ile "fn.grf"+++++++++++++++++++++++++++++
                    otf3.precision(outprec);
otf3<<"\n                    Input File "<<filename.c_str()<<"\n\n" ;

                    otf3<<hdr2 ;
          //     +0.01  -0.881940268668475     -1.78261681790058       -0.572033298124713

                    otf3.setf(ios::showposlios::left);
                    for(k=1;k<=10;k++)
                    {     m=(ninv[k-1]-intercept)/slope;
                              FL=m+g*(m-x_)/(1.-g);
                              D=t/slope/(1.-g)*sqrtl((1.-g)/s0+(m-x_)*(m-x_)/Sxx);
                              otf3.width(5);
                              otf3<<(k*0.01)<<'\t';
                              otf3.width(18);
                              otf3<<PowL(m,lin_log)<<'\t';

                              if(bigslope!=1){
                              otf3.width(18);
                              otf3<<PowL((FL-D),lin_log)<<'\t';
                              otf3.width(18);
                              otf3<<PowL((FL+D),lin_log)<<'\n';}
                              else
                              {
                              otf3.width(18);
                              otf3<<" - "<<'\t';
                              otf3.width(18);
                              otf3<<" - "<<'\n';}
```

```
                };

                for(k=1;k<=16;k++)
                {     m=(ninv[k+9]-intercept)/slope;
                          FL=m+g*(m-x_)/(1.-g);
                          D=t/slope/(1.-g)*sqrtl((1.-g)/s0+(m-x_)*(m-x_)/Sxx);
                          otf3.width(5);
                          otf3<<(.1+k*0.05)<<'\t';
                          otf3.width(18);
                          otf3<<PowL(m,lin_log)<<'\t';
                          if(bigslope!=1){
                          otf3.width(18);
                          otf3<<PowL((FL-D),lin_log)<<'\t';
                          otf3.width(18);
                          otf3<<PowL((FL+D),lin_log)<<'\n';}
                          else
                          {
                          otf3.width(18);
                          otf3<<" - "<<'\t';
                          otf3.width(18);
                          otf3<<" - "<<'\n';}

                };

            for(k=1;k<=9;k++)
                {     m=(ninv[k+25]-intercept)/slope;
                          FL=m+g*(m-x_)/(1.-g);
                          D=t/slope/(1.-g)*sqrtl((1.-g)/s0+(m-x_)*(m-x_)/Sxx);
                          otf3.width(5);
                          otf3<<(.9+k*0.01)<<'\t';
                          otf3.width(18);
                          otf3<<PowL(m,lin_log)<<'\t';
                          if(bigslope!=1){
                          otf3.width(18);
                          otf3<<PowL((FL-D),lin_log)<<'\t';
                          otf3.width(18);
                          otf3<<PowL((FL+D),lin_log)<<'\n';}
                          else
                          {
                          otf3.width(18);
                          otf3<<" - "<<'\t';
                          otf3.width(18);
                          otf3<<" - "<<'\n';}

                };
};
//-------------------------------------------------------------------------------
//++++++++++++++++++++Out put to Main Window++++++++++++++++++++++++++++++++++++++
strcat(fn,"\n    Intercept = ") ;
strcat(fn,gcvt(intercept,21,msg));
strcat(fn,"\n");
strcat(fn,"\n        Slope = ") ;
strcat(fn,gcvt(slope,21,msg));
strcat(fn,"\n");
```

```
if(lin_log==1)
{ strcat(fn,"\n      LogED50 = ") ;
 strcat(fn,gcvt(log_ED50,21,msg));
 strcat(fn,"\n");
};
strcat(fn,"\n          ED50 = ") ;
strcat(fn,gcvt(ED50,21,msg));
strcat(fn,"\n");
strcat(fn,"\n Log Fiducial Limits \n ") ;
strcat(fn,"\n");
strcat(fn,gcvt(FLL,21,msg));
strcat(fn,"          ");
strcat(fn,gcvt(FLU,21,msg));
strcat(fn,"\n");
//---------------------------------------------------------------------------
 otf3.close();
 otf2.close();
otf.close();

FlashWindow(0);

Invalidate();

SetCursor(NULL,IDC_ARROW);

         ofile="notepad "+ofile;
          ofile2="notepad "+ofile2;
                ofile3="notepad "+ofile3;


          //   ofile2="c:\\Excel\\excel c:\\excel\\probit.xlt";       //this works
         WinExec((const char*)ofile.c_str(),SW_RESTORE);
          WinExec((const char*)ofile2.c_str(),SW_RESTORE);
                if(lin_log==1) WinExec((const char*)ofile3.c_str(),SW_RESTORE);
};

}

//
//
void
TCommDlgWnd::CmHelpAbout()
{
 MessageBox("      Finney's Probit  \nWritten by Dr. Lonnie W. Manning\n"
                              "     Copyright (c) 1995  \n         The NutHouse",
                              "About EZ-PROBIT",
                              MB_OK);
}
//-----------------------------------------------------------------

void
TCommDlgWnd::CmEmpInput()
{
 char empInfo[sizeof(TParamStruct)+2+1];
 if (TParamDlg(this,"PARAMINFO", ParamStruct).Execute() == IDOK) {

strcpy(empInfo,ParamStruct.DefaultDir);
        strcat(empInfo, "\n");
        strcat(empInfo, ParamStruct.Converg);
        strcat(empInfo, "\n");
/*      strcat(empInfo, ParamStruct.DeptEdit);
        strcat(empInfo, "\n");
        strcat(empInfo, ParamStruct.SecEdit);
        strcat(empInfo, "\n");                    */
//      strcpy(empInfo, ParamStruct.Log_Dose ? "FullTime " : "PartTime ");
strcat(empInfo, ParamStruct.Log_Dose ? "Log Dose\n " : "Lin Dose\n ");

        strcat(empInfo, ParamStruct.PermFile ? "Save Output\n" : "Don\'t Save Output\n ");
        strcat(empInfo, ParamStruct.Exempt ? "Exempt\n " : "NonExempt\n ");
        MessageBox(empInfo, "Information Stored", MB_OK);
 }
}

//----------------------------------------------------------------
class TCommDlgApp : public TApplication {
 public:
        TCommDlgApp() : TApplication() {}
        void InitMainWindow() {
                MainWindow = new TCommDlgWnd(0, "EZ-PROBIT");
                 MainWindow->SetIcon(this, "EZ_PROBIT");
                EnableCtl3d();
        }
};

int
OwlMain(int /*argc*/, char* /*argv*/ [])
{
 return TCommDlgApp().Run();
}
```